SIMD/MIMD PROCESSING ON A RECONFIGURABLE ARRAY

BACKGROUND OF THE INVENTION

The present invention relates generally to computer processors. More specifically, the present invention relates to a reconfigurable processor architecture for SIMD/MIMD processing.

Single instruction, multiple data (SIMD) processors are a type of processor that executes one instruction on different data items. In an array of processor cells, each cell is configured to execute the same instruction. Data on which the same instruction is executed by multiple cells is known as block data. SIMD processors have relatively simple architectures, and are sufficiently powerful for applications that process block data.

However, SIMD processors are inefficient for applications that require adaptive processing of heterogeneous data. The term heterogeneous data relates to data being processed by multiple processing cells, at least a portion of which execute different instructions. Multiple instruction, multiple data (MIMD) are a type of processor in which several different instructions are executed, for processing different data items. MIMD processors can support heterogeneous applications, but are much more costly and complex to program. Further MIMD processors are more difficult to interface to other MIMD processors for scalability.

What is needed is a processor having the simplicity and efficiency of a SIMD, yet the adaptability and scalability of a MIMD.

SUMMARY OF THE INVENTION

This invention relates to processing architectures and methods of processing data. In an embodiment of the invention, an array of SIMD processor cells includes control and logic circuitry for dynamic reconfiguration of individual cells for MIMD-type functionality.

In one embodiment, an apparatus for programming an MxN array of reconfigurable processor cells, where each cell being operative according to a context instruction, includes an enable register, and an execution mode generator. The enable register is connected to the array and having an enable signal for controlling an active state of selected cells in the array. The execution mode generator is connected to each cell in the array and provides an execution mode signal for controlling delivery of a next context instruction to each active cell in the array based on the enable signal.

In another embodiment of the invention, a SIMD/MIMD processing system includes a plurality of reconfigurable processing cells, wherein a function of each cell is operative according to a context, a controller for controlling an execution mode of the array, wherein each cell of the array executes either a present or an updated context during each clock cycle, and an enable register for selectively enabling a subset of cells in the array for receiving the updated context instruction while all non-enabled cells maintain the present context. In accordance with the embodiment, the internal states of each cell are updated by the context register only when the cell is in an active state, wherein the active state is based on the execution mode.

In yet another embodiment of the invention, a method of programming an MxN array of cells includes enabling a sub-unit of cells for a new context instruction, wherein each non-enabled cell maintains its present context instruction, and providing the new context instruction to each cell in the sub-unit. Alternatively, a method of programming an MxN array of cells includes selecting an execution mode for each cell in the array, executing a present context by each cell having an asserted execution mode bit, and executing an updated context by each cell having an unasserted execution mode bit.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 shows a dynamically reconfigurable processing architecture including an MxN array of reconfigurable cells.

FIG. 2 illustrates an internal structure of one reconfigurable cell according to an embodiment of the invention.

FIG. 3 depicts an MxN array of reconfigurable cells with enable registers.

FIG. 4 depicts a control signal source according to one embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

As described above, processor architectures for (MIMD) processing are normally complex and difficult to interface together. The present invention overcomes the aforementioned limitations of conventional SIMD or MIMD processors by providing an architecture and control structure for performing SIMD/MIMD processing with a reconfigurable array of processing cells.

FIG. 1 shows a dynamically reconfigurable processing structure 100 in accordance with an exemplary embodiment of the invention. The processing structure 100 includes an M row x N column array of reconfigurable cells 110. In the exemplary embodiment, the MxN array is an 8x8 array of cells 110. Each cell 110 is a processor that is configured, and reconfigurable, for performing one or more logic function. Each row M is controlled by a row decoder 120, which is configured to address and instruct all cells in the entire row. Each column N is controlled by a column decoder 130 configured to address and provide instructions to all cells in the entire column.

FIG. 2 illustrates the internal structure of one cell 110 according to the exemplary embodiment of the invention. In the exemplary embodiment, the cell 100 includes one or more functional units 210, 220 and 230. The combination of functional units 210, 220 and 230 defines an operation unit of the cell, and represents a logical function executed by the cell 110. Suitable functional units include, but are not limited to, a Multiply-and-Accumulate (MAC) functional unit, an arithmetic unit, and a logic unit. Other types of functional units for performing functions are possible. The functional units 210, 220 and 230are configured in the cell 110 in a modular fashion, in which other functional units can be added, or present functional units can be removed. In particular, by adding functional units, the range of functions of the cell 110 is

expandable and scalable. The modular design of the exemplary embodiment also makes decoding of the function easier.

In one embodiment of the invention, only one of the functional units 210, 220 or 230 is active at a time during one clock cycle. The functional units are controlled and activated by a context register 240. The context register 240 latches a context instruction upon each processing cycle, and provides the context instruction to the appropriate functional unit(s). In other embodiments, depending upon the structure and logic of the group of functional units, and based on the context of the cell, more than one functional unit can be activated at a time.

Each cell 110 contains a register file 212 for storing temporary data, such as a result of logical computations by the functional units 210, 220 and 230, for example. In the example, the result of each functional unit is combined together by multiplexer 204, output to a shifter 206, and provided to an output register 216. The data output of the shifter 206 is also provided to the register file 212, where it is temporarily stored until replaced by a new set of output data from the combination of functional units 210, 220 and 230. The output register 216 sends the output data to an output multiplexer 218, from which the output data, representing a processing result of the reconfigurable cell, is sent to either the data bus, to a neighboring cell, or both.

An CONTROL1 signal is gated with a clock signal for controlling most or all of the sequential logic elements within the cell 110. The CONTROL1 signal controls the flow of data to be operated upon by the cell 110. An CONTROL2 signal is gated with the clock signal for controlling the context register 240. The CONTROL2 signal controls the flow of a the context instruction to the cell 110 for controlling the operation of the cell 110. The context register 240 stores a next context instruction upon each clock cycle, and provides the next context instruction to the functional units 210, 220 and 230. The CONTROL1 and CONTROL2 signals are based on the enable signals provided by row and column enable registers 310 and 320, respectively, and an execution mode generator 330, described with reference to FIG. 3.

As stated above, a present function of the cell 110 is controlled by a context. The context defines logic and data to be executed by the functional units 210, 220 and 230. The context is broadcast to cells 110 in a row M or column N in the MxN array from a context memory connected to a context register 240. The context memory is placed at the periphery of the array 100. Thus, the processor structure using the array of cells can operate in SIMD mode, i.e. executing the same instruction on different data items. Instructions suitable for SIMD mode include, but are not limited to, matrix and vector operations.

FIG. 3 shows a processor array 300 with configuration control for implementing MIMD functions in a SIMD array, in accordance with an embodiment of the invention. In the embodiment, an execution mode signal generator 330 is gated to all cells in the array to decide the execution mode of the array 300. An M-bit row enable register 310 is connected to each row of the MxN array. An N-bit column enable register 320 is connected to each column of the MxN array. The clock signal of the context register for each cell in the array is gated with the enable signal from each of the row and column enable registers 310, 320 and the execution mode signal for the execution mode generator 330. Depending upon the values in each enable register 310, 320, the context registers 240 of a subset of the cells 110 in the MxN array 300 can receive updated contexts at any given time, thereby forming a "mask" over the array 300. This mask can be updated at each cycle, if needed. For a particular cell to be active, both corresponding bit lines from the row and column enable registers 310 and 320 must carry a logically active signal, such as a logical "1" for example, depending on a desired convention.

When the array 300 is in SIMD execution mode, represented by a "00" state of the execution mode generator 330, each enabled cell 110 receives a new context instruction. The context instruction is loaded into the cell 110 for each subsequent clock cycle so long as the execution mode signal to the cell 110 remains unchanged. Although the context instruction can be dynamically updated every cycle, the cells110 in each row/column preferably execute the same context instruction. When the array 300 is in MIMD execution mode (state '10'), the current context

instruction for each enabled cell 110 is maintained. When the context instruction is maintained, the cell 110 is effectively "frozen" to repeatedly execute that context instruction, until a new context instruction is designated for the cell 110. When the array 300 is in Programming mode (state "01"), the internal states of each cell 110 are not updated by their associated context registers 240, and an updated context can be broadcast to each cell 110 enabled by the enable registers 310, 320. Table 1 summarizes the various execution modes of the array 300.

STATE	EXECUTION MODE
00	Default. Array is in SIMD execution mode. The enable registers are also in the default state, which is 0. Each cell receives new context, and executes on it every clock cycle.
01	Context distribution mode (i.e. "programming mode"). Array is being programmed into an MIMD machine. The enable registers are set to a certain value, and the contexts are broadcast to the selectively enabled cells. In this mode, the internal state of each cell will not be changed.
10	MIMD execution mode. After the array is programmed as a MIMD machine, the enable registers are released and reset, and the execution mode generator is set to "10." The array stops receiving new contexts, and executes on the present contexts.
11	"don't care"

TABLE 1.

By selectively enabling the context registers of a subset of cells 110 in the array, it is possible to broadcast context instructions to each active-masked or enabled cell. Accordingly, by enabling subunits representing one or more cells 110 within the array 300, different portions of the array 300 can be configured to execute different operations. Further, different operations are the result of different functional units being activated by the context instruction, which can occur even within the same row and/or column. It is therefore possible to separately program each individual cell

to perform a different function, yielding MxN different operations. Thus, a reconfigurable cell array 300 according to the invention can be configured as a one-dimensional vector processor, or as a two-dimensional array processor.

The reconfigurable cells in an array are interconnected according to one or more hierarchical schemes. In one exemplary embodiment, for an 8x8 array for example, cells within a quadrant, i.e. each group of 4x4 cells, are fully connected in a row or column. Further, cells in adjacent quadrants are connected via specially-configured fast lanes that enable a cell in one quadrant to broadcast function results to all cells in an adjacent quadrant.

FIG. 4 depicts a configuration for generating the CONTROL1 and CONTROL2 signals using the enable signals and the execution mode signal. The execution mode '01' signal is inverted to produce the CONTROL1 signal. A row enable signal is provided from the row enable register, and combined with a column enable signal provided from the column enable register. The combined enable signal is provided to a AND gate with the execution mode '01' signal, and the output of the AND gate is connected to a NOR gate with the execution mode '10' signal to produce the CONTROL2 signal. Other configurations and methods for producing the CONTROL1 and CONTROL2 signals will be readily apparent to a person of ordinary skill in the art.

Other embodiments, combinations and modifications of this invention will occur readily to those of ordinary skill in the art in view of these teachings. Therefore, this invention is to be limited only be the following claims, which include all such embodiments and modifications when viewed in conjunction with the above specification and accompanying drawings.

WHAT IS CLAIMED IS: